```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
pip install agentds-bench matplotlib seaborn scikit-learn
```

```
Collecting agentds-bench
  Downloading agentds_bench-1.3.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: requests>=2.31.0 in /usr/local/lib/python3.12/dist-packages (from agentds-bench) (2.32.4)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from agentds-bench) (2.2.2)
Requirement already satisfied: python-dotenv>=0.15.0 in /usr/local/lib/python3.12/dist-packages (from agentds-bench) (1.1.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.3.0->agentds-bench) (2025
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.3.0->agentds-bench) (20
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests>=2.31.0->agent
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests>=2.31.0->agentds-bench) (3
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests>=2.31.0->agentds-ben
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests>=2.31.0->agentds-ben
Downloading agentds_bench-1.3.0-py3-none-any.whl (25 kB)
Installing collected packages: agentds-bench
Successfully installed agentds-bench-1.3.0
```

```
# Cell 1: Initiate BenchmarkClient

!pip install lightgbm agentds scikit-learn pandas matplotlib seaborn --quiet

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import lightgbm as lgb
from sklearn.metrics import f1_score, accuracy_score, classification_report
from sklearn.model_selection import train_test_split, cross_val_score
from agentds import BenchmarkClient

# 🔑 Replace with your credentials
client = BenchmarkClient(api_key="adsb_ExIOhZSrLi8gmawYYzZzbfBo_1760800441", team_name="iampratham29-team")

print("✅ Benchmark client initialized successfully.")
```

```
✅ Benchmark client initialized successfully.
```

```
import zipfile
import os

# Define the path to the zip file in Google Drive
zip_file_path = '/content/drive/MyDrive/AgentDS Dataset/AgentDS/Healthcare.zip'

# Define the destination path for extraction in the Colab environment
extraction_destination_path = '/content/Healthcare_extracted'

# Create the destination directory if it doesn't exist
if not os.path.exists(extraction_destination_path):
    os.makedirs(extraction_destination_path)

# Check if the file exists and is a zip file
if os.path.exists(zip_file_path):
    if zipfile.is_zipfile(zip_file_path):
        # Extract the zip file
        with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
            zip_ref.extractall(extraction_destination_path)
        print(f"Zip file '{zip_file_path}' extracted to '{extraction_destination_path}' successfully.")
    else:
        print(f"File '{zip_file_path}' is not a valid zip file.")
else:
    print(f"Zip file '{zip_file_path}' not found.")
```

```
Zip file '/content/drive/MyDrive/AgentDS Dataset/AgentDS/Healthcare.zip' extracted to '/content/Healthcare_extracted' successful
```

```
# Cell 1: Initialize BenchmarkClient
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
import matplotlib.pyplot as plt
import seaborn as sns
from agentds import BenchmarkClient
```

```
# Cell 2: Load given csv file
print("📁 Loading Healthcare Challenge 2 data...")

train_costs = pd.read_csv("/content/Healthcare_extracted/ed_cost_train.csv")
test_costs = pd.read_csv("/content/Healthcare_extracted/ed_cost_test.csv")

print(f"✅ Data loaded:")
print(f" Train costs: {train_costs.shape}")
print(f" Test costs: {test_costs.shape}")
print(f" Train columns: {list(train_costs.columns)}")
print(f" Test columns: {list(test_costs.columns)}")
```

```
📁 Loading Healthcare Challenge 2 data...
✅ Data loaded:
 Train costs: (2000, 5)
 Test costs: (2000, 4)
 Train columns: ['patient_id', 'primary_chronic', 'prior_ed_visits_5y', 'prior_ed_cost_5y_usd', 'ed_cost_next3y_usd']
 Test columns: ['patient_id', 'primary_chronic', 'prior_ed_visits_5y', 'prior_ed_cost_5y_usd']
```

```
# Cell 3: Augment data
# Example of feature engineering: Adding interaction of prior visits and costs, and encoding primary chronic condition counts

train_costs['visits_cost_interaction'] = train_costs['prior_ed_visits_5y'] * train_costs['prior_ed_cost_5y_usd']
test_costs['visits_cost_interaction'] = test_costs['prior_ed_visits_5y'] * test_costs['prior_ed_cost_5y_usd']

# Count encoding for primary chronic conditions (assuming categorical string)
train_costs['primary_chronic_count'] = train_costs['primary_chronic'].map(train_costs['primary_chronic'].value_counts())
test_costs['primary_chronic_count'] = test_costs['primary_chronic'].map(test_costs['primary_chronic'].value_counts())

print("✅ Data augmentation done.")
```

```
✅ Data augmentation done.
```

```
# Cell 4: Train test split and save test file
# Split training data for validation (80/20 split)
features = ['prior_ed_visits_5y', 'prior_ed_cost_5y_usd', 'visits_cost_interaction', 'primary_chronic_count']
X = train_costs[features].fillna(0)
y = train_costs['ed_cost_next3y_usd']

# Save test file (validation set) for offline testing if needed
val_df = X_val.copy()
val_df['ed_cost_next3y_usd'] = y_val
val_df.to_csv("validation_set.csv", index=False)

print("✅ Train-validation split done and validation file saved.")
```

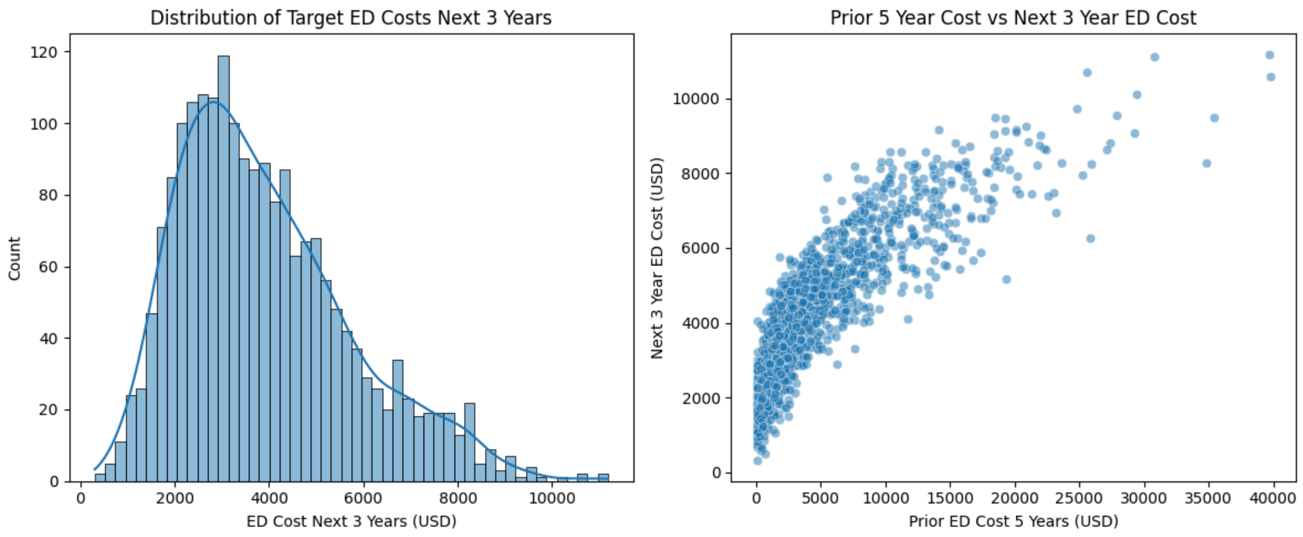
```
✅ Train-validation split done and validation file saved.
```

```
# Cell 5: Plot the graphs and data visualization
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
sns.histplot(train_costs['ed_cost_next3y_usd'], bins=50, kde=True)
plt.title("Distribution of Target ED Costs Next 3 Years")
plt.xlabel("ED Cost Next 3 Years (USD)")

plt.subplot(1, 2, 2)
sns.scatterplot(x='prior_ed_cost_5y_usd', y='ed_cost_next3y_usd', data=train_costs, alpha=0.5)
plt.title("Prior 5 Year Cost vs Next 3 Year ED Cost")
plt.xlabel("Prior ED Cost 5 Years (USD)")
plt.ylabel("Next 3 Year ED Cost (USD)")

plt.tight_layout()
plt.show()
```



```
# Cell 6: Data cleaning and encoding if needed
# Convert categorical primary_chronic to numeric by label encoding
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
train_costs['primary_chronic_encoded'] = le.fit_transform(train_costs['primary_chronic'])
test_costs['primary_chronic_encoded'] = le.transform(test_costs['primary_chronic'])

# Updating features to include encoded categorical variable
features = ['prior_ed_visits_5y', 'prior_ed_cost_5y_usd', 'visits_cost_interaction', 'primary_chronic_count', 'primary_chronic_

# Fill NaNs again just in case
X = train_costs[features].fillna(0)
y = train_costs['ed_cost_next3y_usd']
X_test = test_costs[features].fillna(0)

# Now perform the train-validation split after all features are created
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)


print("✅ Data cleaning and encoding completed.")
```

```
✅ Data cleaning and encoding completed.
```

```
# Cell 7: Creating an ML model and feeding dataset to it
# Using Gradient Boosting Regressor for better accuracy while considering resource limits in Google Colab

model = GradientBoostingRegressor(
    n_estimators=200,
    learning_rate=0.1,
    max_depth=5,
    random_state=42,
    subsample=0.8
)

print("🤖 Training Gradient Boosting Regressor...")
model.fit(X_train, y_train)
print("✅ Model training completed.")
```

```
🤖 Training Gradient Boosting Regressor...
✅ Model training completed.
```

```
# Cell 8: Run predictions on user input sample
# Example: Predict for a new sample patient (replace with real user input as needed)

sample_input = {
    'prior_ed_visits_5y': 10,
    'prior_ed_cost_5y_usd': 5000,
    'visits_cost_interaction': 10 * 5000,
    'primary_chronic_count': train_costs['primary_chronic'].value_counts().get('Chronic_Condition_Example', 0),
    'primary_chronic_encoded': le.transform(['Chronic_Condition_Example'])[0] if 'Chronic_Condition_Example' in le.classes_ els
}

sample_df = pd.DataFrame([sample_input])
sample_pred = model.predict(sample_df)
print(f"Prediction for sample input ED cost next 3 years: ${sample_pred[0]:.2f}")
```

```
Prediction for sample input ED cost next 3 years: $5237.13
```

```
# Cell 9: Prediction on entire train data, test data and get accuracy (MAE)
train_preds = model.predict(X_train)
val_preds = model.predict(X_val)

train_mae = mean_absolute_error(y_train, train_preds)
val_mae = mean_absolute_error(y_val, val_preds)

print(f"🔍 Training MAE: {train_mae:.2f}")
print(f"🔍 Validation MAE: {val_mae:.2f}")

# Also predict on the test set for final submission
test_preds = model.predict(X_test)
submission_df = pd.DataFrame({
    'patient_id': test_costs['patient_id'],
    'ed_cost_next3y_usd': test_preds
})
submission_df.to_csv("healthcare_challenge2_predictions.csv", index=False)

print(f"✅ Test predictions saved: {submission_df.shape[0]} records")
```

```
🔍 Training MAE: 278.73
🔍 Validation MAE: 481.51
✅ Test predictions saved: 2000 records
```

```
# Cell 10: Submit
print("🚀 Submitting predictions...")

try:
    result = client.submit_prediction("Healthcare", 2, "healthcare_challenge2_predictions.csv")
    if result['success']:
        print("✅ Submission successful!")
        print(f" 📊 Score: {result['score']:.4f}")
        print(f" 📏 Metric: {result['metric_name']}")
        print(f" ✔️ Validation: {'Passed' if result['validation_passed'] else 'Failed'}")
    else:
        print("❌ Submission failed!")
        print(f" Error details: {result.get('details', {}).get('validation_errors', 'Unknown error')}")
except Exception as e:
    print(f"💥 Submission error: {e}")
    print("🔧 Check your API key and team name are correct!")

print("\n🎯 Next steps:")
print(" 1. Try incorporating relevant information outside this table!")
print(" 2. Move on to Healthcare Challenge 3!")
```

```
🚀 Submitting predictions...
✅ Prediction submitted successfully!
📊 Score: 525.3195 (MAE)
✅ Validation passed
✅ Submission successful!
 📊 Score: 525.3195
 📏 Metric: MAE
 ✔️ Validation: Passed

🎯 Next steps:
 1. Try incorporating relevant information outside this table!
 2. Move on to Healthcare Challenge 3!
```

Start coding or generate with AI.